A Computationally Universal Field Computer That is Purely Linear

David H. Wolpert
NASA Ames Research Center,
MS 269-1, Moffett Field, CA 94035, USA
dhw@ptolemy.arc.nasa.gov

Bruce J. MacLennan
Computer Science Department,
University of Tennessee, Knoxville, TN 37996-1301
maclennan@cs.utk.edu

Abstract

A *field computer* is a (spatial) continuum-limit neural net (MacLennan 1987). We investigate field computers whose temporal dynamics is also continuum-limit, being governed by an integro-differential equation. We prove that even when they are purely linear, such systems are computationally universal. The "trick" used to get such universal (and therefore in general nonlinear) behavior is quite similar to the way nonlinear macroscopic physics arises from the purely linear microscopic physics of Schrödinger's equation: one *interprets* the system in a non-linear way. In this paper, we show that simply using a unary code for the interpretation suffices. See (Wolpert and MacLennan, 1993) for full details and additional material.

1 A purely linear continuum-limit neural net

The most natural way to extend conventional neural nets to the continuum limit in the set of neurons is to invoke the concept of a "field", that is a real-valued function on an n-dimensional Euclidean space (MacLennan 1987). In this extension the continuum of neurons is indexed by real vectors $\mathbf{r} \in \mathbf{R}^n$, and the activation of a neuron is represented by $\phi_{\mathbf{r}} = \phi(\mathbf{r})$, the field's value at that point. We represent the dynamics of such a field with a function $f : \mathbf{R} \to \Phi(\mathbf{R}^n)$, where $\Phi(\mathbf{R}^n)$, is some convenient space of allowed fields (e.g., $L_2(\mathbf{R}^n)$). So $\phi = f(t_0)$ is interpreted as the state of a continuum-limit neural net at time t_0; $\phi_{\mathbf{r}} = f(t_0, \mathbf{r})$ is the activation value at the time t_0 of the neuron indexed by $\mathbf{r}$.

In this paper we concentrate on fields whose dynamics is purely linear. The dependence of f on t (i.e., the dynamics of the net) is determined by the continuum-limit version of what neural net dynamics would be if there were no sigmoidal non-linearity, i.e., by the continuum-limit version of multiplying by a weight matrix. More precisely, the dynamics is given by the (linear) integro-differential evolution equation,

$$\partial_t f(t, \mathbf{r}) = \int d\mathbf{r}' \, G(\mathbf{r}, \mathbf{r}') f(t, \mathbf{r}'), \tag{1}$$

which we abbreviate as

$$\dot{f}(t) = G f(t). \tag{2}$$

The "weight matrix" of the net corresponds to the kernel ($G \in \Phi(\mathbf{R}^n \times \mathbf{R}^n)$) of this evolution equation. In particular, a kernel that is non-zero for all values of its arguments corresponds to a neural net whose weight matrix is fully (recurrently) connected.

The most natural way of assigning meaning to the distribution $f(t)$ is in terms of its support across $\mathbf{R}^n$. (This corresponds to interpreting the state of a neural net by examining which neurons have activation values exceeding a certain threshold, a scheme usually called the "unary" representation in the neural net literature.) So for example, if the support across $\mathbf{R}^n$ of $f(t_0)$ covers a region Σ_1, then we interpret $f(t_0)$ as having one meaning, whereas if instead it covers a different region Σ_2, we interpret $f(t_0)$ as having some different meaning. The actual values of f across $\mathbf{R}^n$ only matter insofar as they determine the support of f.

In this paper, even the time when output occurs is determined by the support of f: output is signaled when the support covers a predetermined *output-flagging* region of $\mathbf{R}^n$. So for example, if t_2 is the earliest time when the support of f covers the output-flagging region, then the output of the system is determined by the distribution of $f(t_2)$'s support over $\mathbf{R}^n$. With this scheme, in direct analogy to a Turing machine (TM), the amount of time the net runs is a variable, and in general depends on the input values fed into the net (i.e., depends on the field $f(t_0)$).

2 Computational Universality

This section describes how to construct a G such that the evolution equation for $f(t)$, $\dot{f}(t) = Gf(t)$, when combined with the unary representation interpretation of $f(t)$, is computationally equivalent to any particular TM. The fact that the system is defined over an uncountably infinite space ($\mathbf{R}^n$) won't be necessary in this exposition; in fact, to facilitate the analysis the dynamics will be reduced to an evolution equation over a countably infinite subspace of $\mathbf{R}^n$. All proofs can be found in (Wolpert and MacLennan, 1993).

2.1 How to interpret a field as a Turing Machine

First some notational comments are in order. We will work in $\mathbf{R}^5$ (i.e., $n = 5$). However not all five components will be used to specify the state of the TM; some will help with our book-keeping. Bold lower-case letters indicate vectors, and subscripted italic letters indicate components of a vector. Let $\boldsymbol{\rho}$ be any vector in $\mathbf{R}^4$ and z a real-valued scalar; we define $(\boldsymbol{\rho}, z)$ to be the $\mathbf{R}^5$ vector $(\rho_1, \rho_2, \rho_3, \rho_4, z)$. So for example, if the 4-dimensional vector $\boldsymbol{\sigma}$ tells us something of the TM's state at time t, and if we want the 5th component of our corresponding $\mathbf{R}^5$ vector to equal t, then that corresponding vector $\mathbf{r} \in \mathbf{R}^5$ is given by $\mathbf{r} = (\boldsymbol{\sigma}, t)$. For convenience define the m-dimensional Dirac delta functions as follows:

$$\Delta(\mathbf{r}, \mathbf{s}) \equiv \prod_{k=1}^{m} \delta(r_k - s_k), \quad \text{for } \mathbf{r}, \mathbf{s} \in \mathbf{R}^m.$$

(m is implicitly set by the arguments of the Δ.)

The basic idea is to find a kernel G with solution f such that any element in the support of f, $\mathbf{r}$, can be interpreted in the following manner. First, r_5 serves as a system clock; at any particular time t there is only one value of r_5 such that $f(t, \mathbf{r}) \neq 0$, and this value of r_5 is proportional to t, $r_5 = \omega t$. (This clock is necessary to have the dynamics cycle through the various operations making up an iteration of a TM; without this clock embedded in $\mathbf{R}^n$, the dynamics has no way of knowing what TM operation to apply.) Without loss of generality we take $\omega = 1$.

In addition to this restriction on r_5, we want $f(t, \mathbf{r})$ never to be non-zero except for those r_1 through r_4 on the following lattice: $r_4 \in \{0, 1\}$, and $r_1, r_2, r_3 \in \mathbf{Z}^+$. We define $\Lambda \subset \mathbf{R}^4$ to be this lattice:

$$\Lambda = (\mathbf{Z}^+)^3 \times \{0, 1\}.$$

At any particular time t, there will only be one or two of these lattice points in Λ for which $f(t, \mathbf{r}) \neq 0$. These values of r_1 through r_4 for which $f(t)$ is non-zero code for the condition of the TM as follows: r_1 represents head position on the TM's tape, r_2 represents the numerical value on the tape (which for simplicity is assumed to have a finite number of 1's), r_3 represents the internal state of the TM, and r_4 is a buffer label. As t changes, the values of r_1, r_2, r_3, and r_4 for which $f(t, \mathbf{r}) \neq 0$ should change in exact accord with the dynamics of the TM being emulated. In effect, the lattice points represent possible states of our TM emulation, and at any time t, $f(t)$ "points" to a few of these states.

Formally, the goal is to find a G such the evolution equation has a solution with the following properties. First, the solution must be of the form

$$f(t, \mathbf{r}) = \sum_{\boldsymbol{\sigma} \in \Lambda} \Delta[\mathbf{r}, (\boldsymbol{\sigma}, t)] K(t, \boldsymbol{\sigma}). \qquad (3)$$

This solution is a superposition of five-dimensional Dirac delta functions, all centered in r_5 about the point t. Each delta function is centered in $\mathbf{R}^4$ about a different one of the allowed lattice sites $\boldsymbol{\sigma}$, with magnitude $K(t, \boldsymbol{\sigma})$ at each such site. Second, at any time t there must only be a few values of $\boldsymbol{\sigma}$ such that $K(t, \boldsymbol{\sigma}) \neq 0$. It is the dynamics of this support of $K(t, \boldsymbol{\sigma})$ which must correspond to the dynamics of the TM under the interpretation of r_1 through r_4 outlined above. We must construct a G such that the evolution equation has solution of the form Eq. 3, where the coordinate projections of the support of the associated function K obey the dynamics of the TM being emulated. In this way dynamics over $\mathbf{R}^5$ is reduced to dynamics over Λ.

2.2 Reducing to countably infinite dynamics

This subsection shows how to choose a G with solution given by Eq. 3, for arbitrary K. The subsequent subsection shows how to choose K so that the dynamics over Λ emulates an arbitrary Turing machine.

Lemma 1: Let

$$G(\mathbf{r}, \mathbf{r}') = -\partial_{r_5} \Delta(\mathbf{r}, \mathbf{r}') + \delta(r_5 - r_5') \Gamma(\mathbf{r}, \mathbf{r}'), \qquad (4)$$

where for any $\boldsymbol{\rho} \in \mathbf{R}^4$, $\boldsymbol{\sigma} \in \Lambda$, the function Γ obeys

$$\sum_{\boldsymbol{\sigma}} \Gamma((\boldsymbol{\rho}, t), (\boldsymbol{\sigma}, t)) K(t, \boldsymbol{\sigma}) = \sum_{\boldsymbol{\sigma}} \Delta(\boldsymbol{\rho}, \boldsymbol{\sigma}) \partial_t K(t, \boldsymbol{\sigma}) \quad (5)$$

for some time-varying field $K_t \in \Phi(\mathbf{R}^5)$. Then Eq. 3 satisfies the evolution equation (Eq. 1). (The behavior of $\Gamma(\mathbf{r}, \mathbf{r}')$ (and therefore of $G(\mathbf{r}, \mathbf{r}')$) for points $\mathbf{r}'$ whose first four components do not lie on Λ is completely free.)

To use this to reduce dynamics over $\mathbf{R}^5$ to dynamics over Λ, choose $\Gamma(\mathbf{r}, \mathbf{r}') \equiv \sum_{\boldsymbol{\sigma}'' \in \Lambda} \Delta(\boldsymbol{\rho}, \boldsymbol{\sigma}'') A(\mathbf{r}, \mathbf{r}')$ for some function $A \in \Phi(\mathbf{R}^5 \times \mathbf{R}^5)$ ($\boldsymbol{\rho}$ being the vector of the first four components of $\mathbf{r}$). Because of the delta function $\Delta(\boldsymbol{\rho}, \boldsymbol{\sigma}'')$, the terms of this summation are nonzero only when $\boldsymbol{\rho} = \boldsymbol{\sigma}''$. Therefore the $\mathbf{r} = (\boldsymbol{\rho}, t)$ appearing inside $A(\mathbf{r}, \mathbf{r}')$ can be replaced with $(\boldsymbol{\sigma}'', t)$. Hence,

$$\Gamma(\mathbf{r}, \mathbf{r}') = \sum_{\boldsymbol{\sigma}'' \in \Lambda} \Delta(\boldsymbol{\rho}, \boldsymbol{\sigma}'') A[(\boldsymbol{\sigma}'', t), \mathbf{r}'].$$

Substituting this Γ into Eq. 5 yields:

$$\begin{aligned}
&\sum_{\boldsymbol{\sigma}} \Delta(\boldsymbol{\rho}, \boldsymbol{\sigma}) \partial_t K(t, \boldsymbol{\sigma}) \\
&\quad = \sum_{\boldsymbol{\sigma}} \Gamma[(\boldsymbol{\rho}, t), (\boldsymbol{\sigma}, t)] K(t, \boldsymbol{\sigma}) \\
&\quad = \sum_{\boldsymbol{\sigma}''} \left\{ \Delta(\boldsymbol{\rho}, \boldsymbol{\sigma}'') \sum_{\boldsymbol{\sigma}'} A[(\boldsymbol{\sigma}'', t), (\boldsymbol{\sigma}', t)] K(t, \boldsymbol{\sigma}') \right\}.
\end{aligned}$$

This equality can be enforced if individual terms on the right cancel with individual terms on the left, i.e., if

$$\partial_t K(t, \boldsymbol{\sigma}) = \sum_{\boldsymbol{\sigma}'} A[(\boldsymbol{\sigma}, t), (\boldsymbol{\sigma}', t)] K(t, \boldsymbol{\sigma}').$$

Since t is the fifth component of both $(\boldsymbol{\sigma}, t)$ and $(\boldsymbol{\sigma}', t)$, we can re-express the dependence of A on its arguments to get the following:

$$\partial_t K(t, \boldsymbol{\sigma}) = \sum_{\boldsymbol{\sigma}'} A(t, \boldsymbol{\sigma}, \boldsymbol{\sigma}') K(t, \boldsymbol{\sigma}'),$$

which is an "infinite matrix product,"

$$\dot{K}(t) = A(t)\, K(t). \tag{6}$$

This equality is a discrete-space version of the original evolution equation, with one important difference. Whereas the evolution equation had a time-independent kernel G, the equation governing the dynamics of K has a kernel $A(t)$ which depends explicitly on t.

The results so far can be summarized as follows. Choose a function $A(t)$. This function specifies a $\Gamma(\mathbf{r}, \mathbf{r}')$. Now choose a function $K(t)$ which satisfies Eq. 6 at the lattice points. This function specifies an $f(t, \mathbf{r})$. We know that this $\Gamma(\mathbf{r}, \mathbf{r}')$ and this $f(t, \mathbf{r})$ together satisfy Eq. 5. Accordingly, this $f(t, \mathbf{r})$ together with the $G(\mathbf{r}, \mathbf{r}')$ given by $\Gamma(\mathbf{r}, \mathbf{r}')$ jointly satisfy Eq. 1. In other words, so long as we choose an $A(t)$ and a $K(t)$ which jointly satisfy Eq. 6, we will be assured that the $f(t, \mathbf{r})$ based on $K(t)$ satisfies Eq. 1 with a $\Gamma(\mathbf{r}, \mathbf{r}')$ based on $A(t)$. Furthermore, $K(t, \mathbf{r})$ and $f(t, \mathbf{r})$ are non-zero for the exact same $\mathbf{r}$ values, all from within Λ. Therefore the meaning of $f(t, \mathbf{r})$ is given by the Λ-support of the associated $K(t, \sigma)$. So our task is reduced to the following: Given any particular TM, find an $A(t)$ such that the associated $K(t)$ (associated via Eq. 6) has a Λ-support which emulates that TM. The next part of this section describes how to do this.

2.3 Emulating a particular TM

There are several separate logical operations making up an iteration of a TM. It will take the dynamics of the system exactly 1 unit of time to complete each such operation, and $A(t)$ is fixed for each such operation. In other words, if we assume that the system starts evolving at $t = 0$, then $A(t)$ remains unchanged throughout each of the separate intervals $t \in [n, n+1)$, $n = 0, 1, 2, \ldots$.

Four distinct operations occurring in four such consecutive temporal intervals together make up a single iteration of a TM. At the beginning of an iteration the current contents of the TM are stored in the $\sigma_4 = 0$ hyperplane. The first operation clears the buffer hyperplane ($\sigma_4 = 1$). The second operation calculates the condition which the TM being emulated will have at the end of its next iteration and stores the (suitably encoded) result in the $\sigma_4 = 1$ hyperplane. (This calculation is based on the current contents of the $\sigma_4 = 0$ hyperplane). The third operation clears the $\sigma_4 = 0$ hyperplane, and the fourth operation copies the contents of the $\sigma_4 = 1$ hyperplane into the $\sigma_4 = 0$ hyperplane. At the end of this cycle, the dynamics repeats itself: $A(t) = A(t+4)$ for all t. (The detailed description of $A(t)$ together with the proof that it performs as claimed is in (Wolpert and MacLennan, 1993).)

3 Discussion

3.1 The system as a cognitive processor

There are other ways of interpreting a system f evolving according to the equation $\dot{f}(t) = Gf(t)$, in addition to viewing it as a continuum-limit neural net. In particular, such a system can be viewed as a "cognitive processor" operating in (massive) parallel. The idea is to view the value of $f(t, \mathbf{r})$ as the "confidence" one has at time t in the proposition labeled by $\mathbf{r}$. The dynamical evolution of the system is the system trying to determine the answer to a question encoded as $f(t_0, \mathbf{r})$, i.e., trying to determine what proposition to have most confidence in, in response to the proposition encoded in the support of $f(t_0, \mathbf{r})$. This evolution can be viewed as infinite parallel streams of thought, each with different confidence levels, interacting with one another in an attempt to answer the question. (The interaction consists of transferring confidence among the various possible $\mathbf{r}$ values according to the evolution equation.)

This confidence-level interpretation doesn't ascribe meaning only to the support of $f(t)$, but also takes into account the actual values of the field $f(t)$. Nonetheless, one might still wish to flag output by running the system until the support of $f(t)$ covers a pre-determined output-flagging region of $\mathbf{R}^n$. In this context, such output flagging means simply that the system processes a question for as long as it takes for it to determine that it has an answer, which (in the form of $f(t)$, the distribution across $\mathbf{R}^n$ of confidence levels) is signaled when the output is flagged (i.e., when one has non-zero confidence that a decision has been made).

Our evolution equation exactly reflects this interpretation of the dynamics as the transferring of a conserved amount of confidence among propositions $\mathbf{r}$ provided that we can express G as

$$G(\mathbf{r}, \mathbf{r}') = H(\mathbf{r}, \mathbf{r}') - \delta(\mathbf{r}' - \mathbf{r}) \int \mathrm{d}\mathbf{r}'' H(\mathbf{r}'', \mathbf{r}) \qquad (7)$$

for some field $H \in \Phi(\mathbf{R}^n \times \mathbf{R}^n)$. To see this, note that for such a G our evolution equation $\dot{f}(t) = Gf(t)$ (Eq. 1) can be rewritten as

$$\dot{f}(t, \mathbf{r}) = \int \mathrm{d}\mathbf{r}' H(\mathbf{r}, \mathbf{r}') f(t, \mathbf{r}') - \int \mathrm{d}\mathbf{r}'' H(\mathbf{r}'', \mathbf{r}) f(t, \mathbf{r}).$$

The second integral represents loss in confidence in the point $\mathbf{r}$ accompanying H-induced transfer of confidence from $\mathbf{r}$ to other points. The first integral represents gain in confidence in $\mathbf{r}$ due to loss of confidence in the other points. In general H need not be symmetric (i.e., $H(\mathbf{r}', \mathbf{r})$ need not equal $H(\mathbf{r}, \mathbf{r}')$), which means that the dynamics governing loss of confidence in $\mathbf{r}$ need not be the same as that governing gain in confidence in $\mathbf{r}$. Whether or not H is symmetric, $f(t)$ maintains normalization through time: $\partial_t[\int \mathrm{d}\mathbf{r} f(t, \mathbf{r})] = 0$, so that our total confidence remains unchanged.

3.2 Schrödinger's equation and computational universality

If we work in $\mathbf{R}^3$ (i.e., if $n = 3$), and if f is allowed to be complex-valued, then for

$$G(\mathbf{r}, \mathbf{r}') = \left(\frac{ih}{2\pi}\right)^{-1} \Big[\ \left(\tfrac{-h^2}{8m\pi^2} \textstyle\sum_{i=1}^3 \tfrac{\partial^2}{\partial^2_{r'_i}} \delta(r_i - r'_i) \right) + \ \delta(\mathbf{r} - \mathbf{r}') V(\mathbf{r}') \ \Big] ,$$

f's evolution equation reduces to Schrödinger's equation for a quantum-mechanical wave-function f:

$$\frac{ih}{2\pi} \partial_t f = -\frac{h^2 \nabla^2 f}{8m\pi^2} + Vf,$$

where as usual V is the potential governing the wave function's evolution and h is Planck's constant.

Now any system purporting to be an (approximation of) a real-world TM is built of components which are, ultimately, quantum mechanical in nature, and in quantum mechanics meaning is ascribed to the support of the wave function f (for sufficiently peaked wave functions). Therefore we can immediately conclude that, via appropriate choice of the potential V, our evolution equation allows TM solutions. (Strictly speaking, this correspondence between our evolution equation and quantum mechanics actually requires that Schrödinger's equation for a set of more than one interacting particles be simulated.) Alternatively, one can establish universality of our evolution equation for compolex-valued fields simply by noting that there exists a G such that that equation emulates Schrödinger's equation, and by then appealing directly to results in quantum Turing machine theory.

3.3 Training

For the system considered in this paper, training the system (i.e., finding a set of weights so that the net reproduces a particular training set) amounts to finding a G such that when $f(0)$ corresponds to one of the inputs in the training set, then the signaled output $f(t$ when output signaling occurs) codes for the corresponding output. Finding such a G is an ill-posed problem, of course; in any scheme for "training" G to reproduce a training set, some sort of regularizer is needed to uniquely fix G. Otherwise one could simply use the preceding several sections to build an infinite number of distinct TM's, all of which reproduce the training set. As a particular example, to choose among the infinity of TM's that reproduce any provided training set, some bias in favor of smaller TM's (perhaps based on an approximation to the algorithmic information complexity of the TM) is a natural choice of regularizer.

When G is not restricted to the form given in Section 2, the need for regularization is even more pronounced. In addition, without that restriction regularization biases like algorithmic complexity become meaningless. In such cases alternative biases must be considered, like those involving integrated curvature of G.

As a final dropping of restrictions, consider regularization in the case of complex-valued fields. The correspondence of our evolution equation for such fields with quantum mechanics means that we can recast the problem of finding a G to reproduce a provided training set as the problem of finding a potential which evolves one set of wavefunctions into another set of wavefunctions. This is nothing other than a quantum mechanical scattering problem! Such problems have been studied intensively for decades. Exploiting this, one way to find a G to reproduce a provided training set (i.e., provided scattering data) is to assume that there are a discrete number of scattering objects and solve for their positions (just as in X-ray diffraction). Regularization in this context could be a bias towards fewer scattering centers, for example.

4 References

MacLennan, B. J. (1987). Technology-independent design of neurocomputers: The universal field computer. In M. Caudill & C. Butler (Eds.), *Proceedings, IEEE First International Conference on Neural Networks* (Vol. 3, pp. 39–49). New York, NY: Institute of Electrical and Electronic Engineers.

Wolpert, D. H. & MacLennan, B. J. (1993). A Computationally Universal Field Computer That is Purely Linear, *SFI TR 93-09-056*, **submitted**.